AMENDMENTS TO THE CLAIMS

1 – 27. (Canceled)

28. (Currently Amended) A computer-implemented method of upgrading a software application by merging two modified versions of a common ancestor version, the method comprising:

receiving indications of a first modified version of the common ancestor version and of a distinct second modified version of the common ancestor version, ~~the first modified version including multiple user customizations~~; and

without user intervention, automatically creating a new ~~upgrade~~ version of the software application by merging the first and second modified versions, by

automatically collecting object definitions from an ancestor repository associated with the common ancestor version, from a first repository associated with the first modified version, and from a second repository associated with the second modified version;

identifying a first set of modified objects ~~user customizations~~ in the first modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions ~~that are not present in the second modified version~~;

identifying a second set of modified objects in the second modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

determining which of the identified modified objects in the first set of modified objects ~~user customizations~~ are compatible with the second modified version; and

creating the new ~~upgrade~~ version of the software application by merging the compatible modified objects ~~user customizations~~ with the second modified version.

29. (Currently Amended) The computer-implemented method of claim 28 further comprising determining which of the identified modified objects in the first set of modified objects are conflicting modified objects with the second modified version ~~wherein the identifying of the user customizations in the first modified version that are not present in the second modified version includes comparing objects resident in the first and second modified versions~~.

30. (Currently Amended) The computer-implemented method of claim 29 ~~28~~, further comprising:

~~determining identified user customizations that conflict with the second modified version;~~

automatically notifying a user of the conflicting modified objects ~~user customizations~~; and

presenting options to the user for resolving the conflicting modified objects ~~user customizations~~.

31. (Cancelled)

32. (Cancelled)

33. (Currently Amended) The computer-implemented method of claim 29 ~~32~~ including automatically resolving one or more of the conflicting modified objects ~~differences~~ without user intervention based at least in part on predefined conflict resolution rules, and wherein the automatic determining of the compatible modified objects ~~differences~~ includes selecting the resolved conflicting modified objects ~~differences~~.

34. (Cancelled)

35. (Cancelled)

36. (Currently amended) The computer-implemented method of claim 30 ~~34~~ including determining an importance of each of one or more of the conflicting modified objects ~~differences~~, and wherein the notifying of the user of conflicting modified objects ~~differences~~ includes providing notification of the determined importances.

37. (Cancelled)

38. (Cancelled)

39. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ wherein the modified objects ~~differences~~ determined to be compatible include conflicting modified objects ~~differences~~ that are identified as superficial.

40. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ wherein the modified objects ~~differences~~ determined to be compatible include conflicting modified objects ~~differences~~ ~~corresponding to objects that are not~~ identified as being of high priority.

41. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ including, before identifying the first set of modified objects and the second set of modified objects ~~the automatic determining of the set of differences~~, displaying to a user a graphical user interface related to creating ~~a~~ the new version of the software application, and wherein the identifying the first set of modified objects and the second set of modified objects ~~automatic determining of the set of differences~~ is in response to an indication from the user via the graphical user interface.

42. (Currently Amended) The computer-implemented method of claim 41 wherein the displaying of the graphical user interface to the user includes displaying an option to include differences between the first and second modified versions ~~in the determined set~~ only if the differences correspond to high priority aspects of the first and second modified versions and/or displaying an option to abort creating ~~a~~ the new version if a specified number of errors is exceeded.

43. (Currently Amended) The computer-implemented method of claim 41 ~~wherein the first and second modified versions of the software application have a common ancestor version of he software application,~~ wherein one of the modified versions has been customized by ~~a~~ the user so as to delete one or more aspects of the common ancestor version, wherein the aspects of the common ancestor version deleted in the one modified version have not been deleted in the other modified version, and including displaying an option to the user to select the deleted aspects as being a compatible difference such that the created new version does not include the deleted aspects.

44. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ including displaying to a user at least some of the differences between each of the two modified versions and the common ancestor version ~~of the determined set~~ so as to allow visual comparison of those two modified versions.

45. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ wherein at least one of the modified versions includes modifications made by a user and/or at least one of the modified versions includes modifications made by a developer of the software application.

46. (Cancelled)

47. (Currently Amended) The computer-implemented method of claim 28 ~~31~~ wherein the first modified version includes user modifications to a prior version of the software application, wherein the second modified version is an upgrade for that prior version, and wherein the creating of the new version is to upgrade the software application to a version of the upgrade that includes at least some of the user modifications.

48. (Cancelled)

49. (Currently Amended) The computer-implemented method of claim 47 ~~48~~, wherein the user-modified prior version comprises a first plurality of objects and the prior version comprises a third ~~second~~ plurality of objects and the upgrade version comprises a second ~~third~~ plurality of objects, and wherein the determining of the first and second sets of modified objects ~~differences~~ comprises:

determining one or more objects from the first plurality of objects that share a common name with one or more objects from the third ~~second~~ plurality of objects;

for each of at least one of the objects from the first plurality of objects that shares a common name with one of the objects from the third ~~second~~ plurality of objects, determining one or more attributes associated with that object that differs from the attributes associated with the object that shares the common name and including data related to the difference between those attributes in the first set of modified objects ~~differences~~;

determining one or more objects from the second plurality of objects that share a common name with one or more objects from the third plurality of objects; and

for each of at least one of the objects from the second plurality of objects that shares a common name with one of the objects from the third plurality of objects, determining one or more attributes associated with that object that differs from the attributes associated with the object that shares the common name and including data related to the difference between those attributes in the second set of modified objects ~~differences~~.

50. (Currently Amended) The computer-implemented method of claim 47 wherein the user-modified prior version comprises a first plurality of objects, the prior version comprises a third ~~second~~ plurality of objects, and the upgrade version comprises a second ~~third~~ plurality of objects, and wherein the determining of the modified objects ~~differences~~ comprises determining that an object from the first plurality of objects is not included within the second or third plurality of objects and indicating that that object is a compatible modified object ~~differences~~.

51. (Currently Amended) The computer-implemented method of claim 47 wherein the user-modified prior version comprises a first plurality of objects, the prior version comprises a third ~~second~~ plurality of objects, and the upgrade version comprises a second ~~third~~ plurality of objects, and wherein the determining of the set of differences comprises determining that an object from the second and third plurality of objects is not included within the first plurality of objects and indicating that absence of that object from the first plurality of objects is a conflicting modified object ~~differences~~.

52. (Currently Amended) A computer-readable medium whose contents cause a computing device to perform a method for upgrading at least portions of a modified version of a software application based at least in part on another distinct modified version of the software application, the method comprising:

~~automatically determining a set of differences between a first modified version of a software application and a second modified version of the software application, the differences in the determined set including at least portions of the first modified version that differ from at least portions of the second modified version;~~

~~automatically determining differences in the determined set that are compatible; and~~

~~upgrading the first second version by including at least some of the compatible differences~~.

automatically collecting object definitions from an ancestor repository associated with the common ancestor version, from a first repository associated with the first modified version, and from a second repository associated with the second modified version;

identifying a first set modified objects in the first modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

identifying a second set of modified objects in the second modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

determining which of the identified modified objects in the first set of modified objects are compatible with the second modified version; and

creating the new version of the software application by merging the compatible modified objects with the second modified version.

53. (Currently Amended) The computer-readable medium of claim 52 ~~wherein the first and second modified versions each include associated objects, and~~ wherein the portions of the first modified version that differ from the portions of the second modified version are objects associated with the first modified version that differ from objects associated with the second modified version.

54. (Cancelled)

55. (Previously Presented) The computer-readable medium of claim 52 wherein the contents that cause the computing device to perform the method is executable software code.

56. (Currently Amended) An apparatus for upgrading a software application from a user-modified prior version to an upgrade version, the user-modified prior version and the upgrade version having a common ancestor version, the apparatus comprising:

~~a first component able to determine a set of differences between the user-modified prior version and at least one of the upgrade version and the common ancestor version, the determining based on a comparison of the user-modified prior version to the upgrade version and/or the common ancestor version;~~

~~a second component able to identify differences in the determined set that are compatible with the upgrade version; and~~

~~a third component able to apply changes to the upgrade version based on the compatible differences.~~

a first component able to automatically collect object definitions from an ancestor repository associated with the common ancestor version, from a first repository associated with the first modified version, and from a second repository associated with the second modified version;

a second component able to identify a first set modified objects in the first modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions and to identify a second set of modified objects in the second modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

a third component able to determine which of the identified modified objects in the first set of modified objects are compatible with the second modified version; and a fourth component able to create the new version of the software application by merging the compatible modified objects with the second modified version.

57. (Currently Amended) A computing device for upgrading a software application from a user-modified prior version to an upgrade version, wherein the user modified prior version and the upgrade version have a common ancestor version, said apparatus comprising:

~~means for determining a first set of differences based on a comparison of the user modified prior version and the common ancestor version and for determining a second set of differences based on a comparison of the upgrade version and the common ancestor version;~~

~~means for determining which differences from said first and second sets of differences are compatible differences and which are conflicting differences; and~~

~~means for applying changes to the upgrade version associated with said compatible differences.~~

means for automatically collecting object definitions from an ancestor repository associated with the common ancestor version, from a first repository associated with the first modified version, and from a second repository associated with the second modified version;

means for identifying a first set modified objects in the first modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

means for identifying a second set of modified objects in the second modified version by comparing against attributes of corresponding objects in the common ancestor version based on the object definitions;

means for determining which of the identified modified objects in the first set of modified objects are compatible with the second modified version; and means for creating the new version of the software application by merging the compatible modified objects with the second modified version.

58-61 (Cancelled)

62-77 (Cancelled)